什么是conda?

Conda 是一个开源的包管理系统和环境管理系统，主要用于 Python 语言，但也可以用于其他语言的项目。

为什么要使用conda?

- 多环境共存，多个环境相互隔离
- 方便安装和管理包依赖
- 支持多平台，如 Windows、Mac OS 和 Linux

下载 conda

conda下载链接

Anaconda和Miniconda区别

特性	Anaconda	Miniconda
安装包大小	大（大约 3GB）	小（大约 50MB）
预安装的包	包含了许多常用的科学计算和数据科学包	只有 `conda` 和最基础的 Python 环境
灵活性	适合初学者，开箱即用	更灵活，适合需要自定义包和工具的用户
包管理工具	`conda`（同 Miniconda）	`conda`（同 Anaconda）
安装过程	直接下载并安装所有工具和库	安装后需要手动安装所需的包

添加到系统变量

这里以Miniconda为例，在Windows 10系统配置环境变量

1、打开环境变量：

- 在 **Windows 10** 和 **Windows 11** 中，右键点击 **此电脑** 或 **计算机**，然后选择 **属性**。
- 点击左侧的 **高级系统设置**。
- 在弹出的窗口中，点击 **环境变量** 按钮。

2、编辑Path变量

- 在 **系统变量** 部分，找到 `Path` 变量，选择它并点击 **编辑**。
- 点击 **新建**，然后添加以下两个路径：

- `D:\miniconda3\Scripts`
- `D:\miniconda3`

3、验证是否添加成功：

```
C:\Users\Administrator>conda --version
conda 25.1.1
```

常用操作

创建新环境

使用 `conda create` 命令来创建一个新的虚拟环境。例如，创建一个 Python 3.8 环境：

```
conda create --name myenv python=3.8
```

这个命令将会创建一个名为 `myenv` 的新环境，并安装 Python 3.8。

你也可以同时指定其他依赖包（例如，安装 `numpy` 和 `pandas`）：

```
conda create --name myenv python=3.8 numpy pandas
```

激活环境

创建环境后，可以使用 `conda activate` 命令激活该环境：

```
conda activate myenv
```

激活后，你的终端会显示环境名称（如 `(myenv)`），表示你现在在该环境中。

查看所有已创建的环境

```
conda env list
```

这会列出所有的环境及其路径。

查看当前激活的环境

可以使用以下命令：

```
conda env list
```

或者：

```
conda info
```

安装或更新包

在激活环境后，你可以安装或更新包。例如，安装 `requests` 包：

```
conda install requests
```

或者：

```
pip install requests
```

要安装其他包，也可以在激活的环境中使用类似的命令。

切换环境

如果你想切换到其他环境，只需使用 `conda activate` 命令：

```
conda activate another_env
```

退出环境

如果你想退出当前的环境，使用：

```
conda deactivate
```

这会将你带回到 base 环境或者默认系统环境。

删除环境

如果你不再需要某个环境，可以通过以下命令删除：

```
conda remove --name myenv --all
```

导出和重建环境

方式一：

如果你希望在其他机器上共享或重建环境，可以导出当前激活的环境配置：

```
conda list --export > environment.txt
```

在其他机器上，可以使用该文件重建环境：

```
conda create --name newenv --file environment.txt
```

只是一个简单的包列表导出工具，适合快速备份或查看当前环境的包信息，但缺乏完整的环境描述。

方式二：

激活你要导出的环境：

```
conda activate trellis  # 替换为你要导出的环境名
```

导出环境配置：

```
conda env export > environment.yml
```

重建环境：

```
conda env create -f environment.yml
```

是一个更完整的环境导出流程，能够生成包含所有必要信息的 YAML 文件，适合用于环境重建和分享。

conda 环境中执行 Python 脚本

激活对应的conda环境然后运行脚本

```
conda activate env1  # 激活 env1 环境
python script1.py  # 在 env1 中执行 script1.py

conda activate env2  # 激活 env2 环境
python script2.py  # 在 env2 中执行 script2.py
```

在命令行中直接指定环境运行脚本

```
conda run -n env1 python script1.py  # 在 env1 环境中运行 script1.py
conda run -n env2 python script2.py  # 在 env2 环境中运行 script2.py
conda run -n 310 set PYTHONIOENCODING=utf-8 && python script1.py
```

`conda run` 是一种方便的工具，适合在不激活环境的情况下临时运行命令，但可能存在兼容性问题，尤其是在复杂环境或需要额外依赖的情况下。

在脚本中使用不同的环境

如果你需要在一个脚本中根据不同的需求运行不同的 Python 代码（例如，调用不同环境的包），你可以在脚本中利用 `subprocess` 模块启动外部进程来运行不同环境的 Python 脚本。

示例：

假设你有两个脚本 `script1.py` 和 `script2.py`，分别需要在不同环境中运行，你可以在 `main_script.py` 中通过 `subprocess` 模块来启动外部进程，指定不同的 conda 环境来执行它们。

```
import subprocess

# 在 env1 环境中运行 script1.py
subprocess.run(['conda', 'run', '-n', 'env1', 'python', 'script1.py'])

# 在 env2 环境中运行 script2.py
subprocess.run(['conda', 'run', '-n', 'env2', 'python', 'script2.py'])
```

这样，`main_script.py` 就会在不同的环境中执行对应的脚本。